```
readme for sum2n

describes design and implementation for sum2n

I basically wrote the full program in C, got it to work, then split the sumInts
function out to a seperate file, and compiled it to assembly. This was originally done
on a 64-bit machine running OS X, so the file was composed largely of gibberish. Doing
this again on a 32-bit ilab machine produced readable assembly, but it very obviously
looked "mechanically constructed", with the compiler choosing odd locations in memory
for variables, etc. Also, I initially tried to use a recursive algorithm, which was
very complex in x86. I re-wrote by hand a new assebmly file based on an iterative
algorithm in C, which actually proved much easier than trying to re-write code produced
by the GCC compiler.

detail design, design/implementation challenges
The c file handles input and stuff like telling the user the proper app usage, and the
assembly file handles the computation of sum2n.
The assembly file is broken into four sub-functions: fib, loop, overflow, and end. Fib
sets up variables from input, loop runs the iterative fibonacci algorithm and detects
overflow, overflow is jumped to when overflow is detected, and puts a -1 to be
returned, and end returns the value found.

analysis of space and time performance of program

Most of the operations are performed once in constant time, but the loop runs a number
of times based on the value supplied to the progam n-1 times, so the order of the
runtime is n.
```